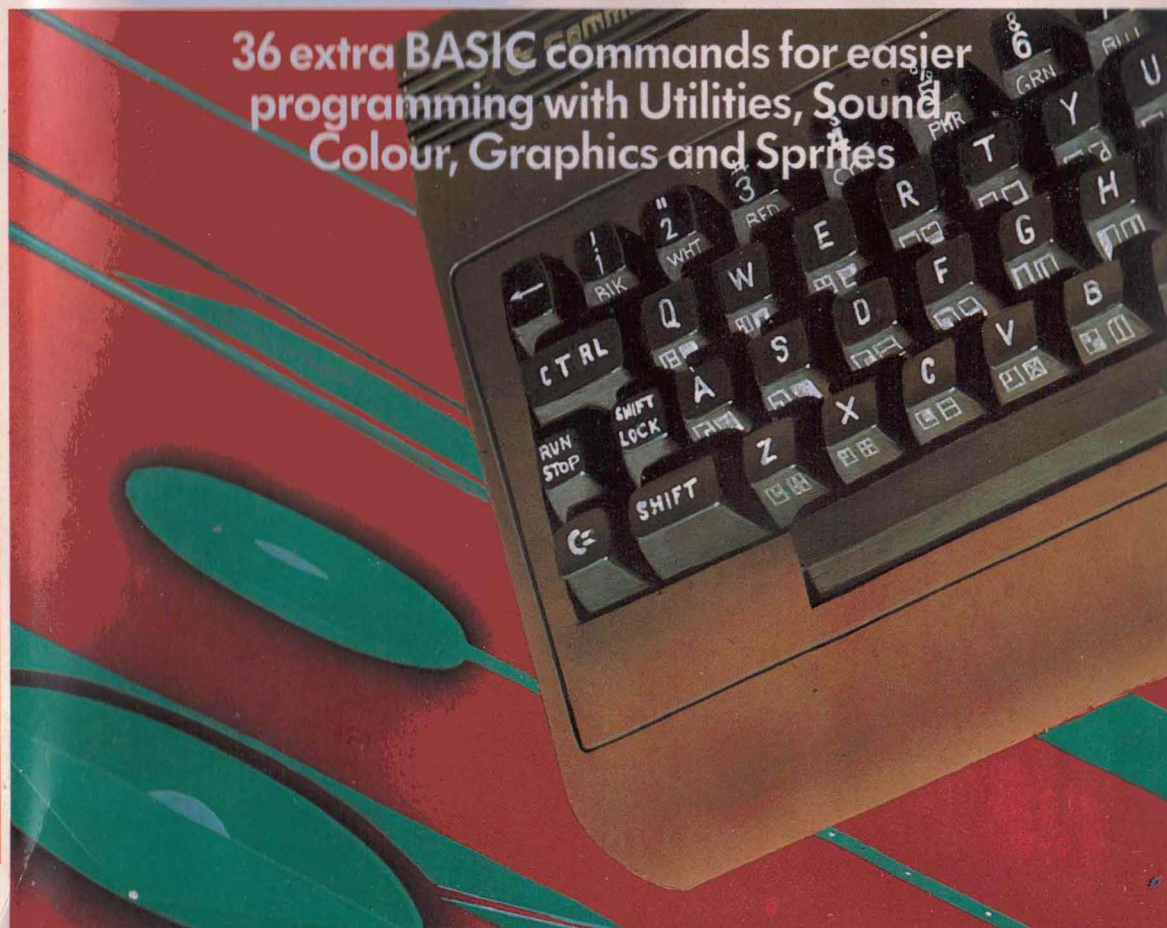
CENTURY SOFTWARE

PETER LUPTON AND FRAZER ROBINSON

SUPER BASIC FOR THE COMMODORE 64

36 extra BASIC commands for easier programming with Utilities, Sound, Colour, Graphics and Sprites



SUPER BASIC FOR THE COMMODORE 64

PETER LUPTON AND FRAZER ROBINSON

SUPER BASIC FOR THE COMMODORE 64

BY

PETER LUPTON & FRAZER ROBINSON

CENTURY COMMUNICATIONS

LONDON

ALSO AVAILABLE FROM CENTURY COMMUNICATIONS:

The Commodore 64 Handbook
The Advanced Commodore 64 Handbook

both by Peter Lupton and Frazer Robinson

Copyright © Peter Lupton and Frazer Robinson 1984

All rights reserved

First published in Great Britain in 1984
by Century Communications Ltd.
12-13 Greek Street,
London W1V 5LE

ISBN 0 7126 0666 1

Typeset by Spokesman, Bracknell

Printed in Great Britain by
Redwood Burn Limited
Trowbridge

NOTE: The programs in this book have been typeset directly from the Commodore 64.

INTRODUCTION

SUPER BASIC is a package of machine code routines which adds 36 new BASIC commands to those provided with the Commodore 64. The range of commands has been carefully chosen to enable the user to make maximum use of the extensive sound, graphics and sprite capabilities of the 64 and to ease the task of programming in BASIC by providing several useful utilities.

This booklet describes the commands in detail and illustrates their use with many example programs.

LOADING INSTRUCTIONS

CASSETTE Switch on the computer and insert the cassette in the cassette unit. Hold down the SHIFT key and press the RUN/STOP key.

DISK Switch on the computer and then the 1541 disk unit. Insert the disk and type:

`LOAD"*",8` and press RETURN

When the "`READY.`" prompt shows, type:

`RUN` and press RETURN.

SUPER BASIC is installed when the loader screen is replaced by a blue screen displaying the message:

SUPER BASIC

(C) PJ LUPTON & FH ROBINSON 1984

64K RAM SYSTEM 38911 BASIC BYTES FREE

WARNING

SUPER BASIC uses all the 4k block of memory between $C000 and $CFFF, and makes extensive use of the spare memory locations in page 0 and page 2 of the RAM. Any program using these areas of memory may corrupt SUPER BASIC.

CONTENTS

LOADING INSTRUCTIONS	
SUPER BASIC UTILITIES	1
SOUND COMMANDS	9
GRAPHICS COMMANDS	22
SPRITE COMMANDS	34
COMMAND SUMMARY	41

SUPER BASIC UTILITIES

SUPER BASIC includes 12 extra utilities which make programming in BASIC easier and more efficient. This section describes these commands and shows how each is used. The first five commands are used only in immediate mode and *not* within programs.

AUTO

The **AUTO** command causes the 64 to automatically generate line numbers as you are entering a program. You specify the line number at which you want to start and the increment, or step, which should be used between the numbers. The syntax is:

```
AUTO Start, Increment
```

For example if you type:

```
AUTO 100,10  (and press RETURN)
```

the 64 will respond by displaying the number 100 leaving the cursor ready for you to enter a program line. When you press RETURN at the end of the program line, the 64 will print the number 110 and so on.

You can specify only a start number, in which case the 64 will use a step of 10, or you can just type **AUTO**, and the 64 will start with line 100 and use a step of 10.

You can tell the 64 to stop generating line numbers by pressing RETURN when a line number is displayed (i.e. entering a blank line). You can

also override the automatically generated numbers by using the cursor keys to move the cursor over them and type in your own. If you subsequently type **AUTO** without any parameters, the 64 will use the most recent values for Start and Increment.

DEL

The **DEL** command allows you to delete blocks of line numbers from a BASIC program and saves you having to type in the line number of each line you wish to delete. The syntax is:

```
DEL Start, End
```

For example if you type:

```
DEL 1000, 2000
```

Every line number between 1000 and 2000 (inclusive) in your BASIC program will be deleted.

If one or both of the line numbers specified in the **DEL** command doesn't exist in your program or you type the line numbers in the wrong order the 64 will display the message:

```
UNDEF'D STATEMENT ERROR
```

and the **DEL** operation will be cancelled.

Once you have **DEL**eted any lines they are gone for ever so ***BE CAREFUL!!***

RENUM

The **RENUM** command allows you to renumber the BASIC program currently in memory. It will renumber all branch instructions - i.e. **GOTO**s,

GOSUBs and **THEN**s, as well as the line numbers. The syntax is:

```
RENUM Start, Increment
```

Any references to a non-existent line number in a branch instruction will be changed to 0 so you will be able to find them easily.

If you type **RENUM** with no parameters, the 64 will assume the default values of 100 for the first line number and 10 for the increment.

If the Start and Increment you specify would cause a line number greater than 63999 to be generated, the 64 will display the message:

```
CAN'T CONTINUE ERROR
```

and the **RENUM** operation will be aborted leaving the program unaltered.

OLD

The **OLD** command allows you to rescue a program after you have accidentally typed **NEW**. It will restore a BASIC program intact provide you have not:

Entered or edited a program line,
Caused a syntax error to occur, or
Defined a BASIC variable.

To use, type **OLD** (RETURN) and the **NEW**ed program will be retored.

PACK

The **PACK** command has no parameters and is used to delete all surplus spaces (except those in quotes) and all **REM** statements from a BASIC program.

This will reduce the amount of space occupied by the program and increase the speed at which it runs, at the expense of legibilty, so it should only be used after you have saved a fully commented version of your program for future reference.

To use, type **PACK** (RETURN) and all surplus spaces and **REM**s will be removed from the program currently in memory.

WARNING If you **PACK** programs with **GOTO**s, **GOSUB**s or other branches to lines with only **REM**s on them, the programs won't work afterwards!

HIMEM

The **HIMEM** command allows you to protect an area of memory from use by BASIC. This gives you a safe area in which to put your machine code routines or data. Its syntax is:

```
HIMEM  Decimal Address
```

For example if you typed:

```
HIMEM 20000
```

all the RAM from address 20000 upward would be protected from BASIC.

DOKE

The **DOKE** command allows you to perform a '16-bit **POKE**' command. The syntax is:

```
DOKE Address, Data
```

where the data is a number between 0 and 65535. The data is stored with the low byte in the

specified address and the high byte in the next location.

For example if you typed:

```
DOKE 1024, 300
```

44 would be stored in location 1024 (this being the low byte of the 16 bit number) and a 1 would be stored in location 1025 representing the high byte. (These locations are at the start of the text screen.)

DEEK

The **DEEK** command is the corresponding command to **DOKE** in that it performs a '16-bit **PEEK**' operation. The syntax is the same as that of the **PEEK** command:

```
PRINT DEEK (Address)
```

or `A = DEEK (B+C/2)`

So:

```
PRINT DEEK (1024)
```

would return the value 300 (ie `PEEK(1024) + 256*PEEK(1025)`) if you had typed in the **DOKE** command example above.

AT

The **AT** command allows you to place the cursor at any position on the screen ready for a **PRINT** or **INPUT** command. The syntax is:

```
AT X coordinate, Y coordinate
```

Since the Commodore screen has 25 rows of 40 columns (numbered 0-24 and 0-39), the allowed range of values for the **AT** parameters is:

X coordinate	0 to 24
Y coordinate	0 to 39

If these parameters are exceeded, the program will stop and the 64 will display the message:

```
'ILLEGAL QUANTITY ERROR IN XXXX'
```

The following short program illustrates the use of the **AT** command by drawing a circle on the text screen:

```
100   PRINT "{CLS}":REM CLEAR SCREEN
110   AT 10,2:INPUT "RADIUS (2 -
      12)";R
120   IF R<2 OR R>12 THEN 100
130   AT 10,2:PRINT "{ 12 SPACES }":
      REM BLANK OUT INPUT FROM LINE
      110
140   FOR J=0 TO 2*π STEP 0.1
150   X = R * SIN(J)+20
160   Y = R * COS(J)+12
170   AT X,Y:PRINT "●":REM CHR$(209)
180   NEXT J
190   GET K$:IF K$="" THEN 190
```

PAUSE

The **PAUSE** command allows you to introduce timed delays into your programs without the need for large **FOR ... NEXT** loops. The syntax is:

```
PAUSE Delay
```

where the Delay parameter is a number between 0 and 65535 representing the delay in Jiffies. A Jiffy is Commodore slang for 1/60th of a second, so the

PAUSE command gives a range of delays from 1/60th of a second to more than 18 minutes. For example the statement:

```
500   PAUSE 300
```

would insert a delay of 5 seconds into a program at line 500.

POP

The **POP** command allows you to exit from a series of nested subroutines neatly, without having to **RETURN** up through each routine. It has no parameters and is useful if, for example, you want to return from several layers of subroutine to report an error. If you use POP when not in a subroutine, the 64 will display the message:

```
'NOT IN SUBROUTINE ERROR IN XXXX'
```

RESET

The final utility command is **RESET** which deactivates SUPER BASIC and performs a switch-on reset of the 64.

SUMMARY - UTILITIES

AUTO Automatic line numbering from a specified line number with a given increment.

```
AUTO Start, Increment
```

DEL Block deletes lines from BASIC program between, and including, specified lines.

```
DEL Start, End
```

RENUM Renumbers BASIC programs from a specified start with specified increment. All branch instructions are redirected.

```
RENUM Start, Increment
```

OLD Restores accidentally **NEW**ed programs.

```
OLD
```

PACK Removes all surplus spaces and all **REM**s from a program.

```
PACK
```

HIMEM Reserves an area at the top of memory for machine code programs and data.

```
HIMEM Address
```

DOKE Performs a 16-bit **POKE** operation.

```
DOKE Address, Data
```

DEEK Performs a 16-bit **PEEK** operation.

```
PRINT DEEK (Address)
```

AT Positions cursor on text screen.

```
PUT X coord, Y coord
```

PAUSE Allows a timed delay in Jiffies (1/60th of a second).

```
PAUSE Jiffies
```

POP Removes one subroutine return address from the stack.

```
POP
```

RESET Resets the 64, de-activates SUPER BASIC and frees BASIC memory.

```
RESET
```

SOUND COMMANDS

INTRODUCTION

The 64's built in sound synthesiser chip – the SID chip – allows an almost infinite range of musical (and non musical !) sounds to be produced on three independent voices. SUPER BASIC adds 7 new commands which provide control over the SID chip, and make programming even complex tunes very easy.

USING THE COMMANDS

VOLUME

Before any sound can be generated, the volume must be set using the **VOLUME** command. The volume command requires *one* parameter, a number between 0 and 15, with 0 meaning silence and 15 representing maximum volume.For example try typing:

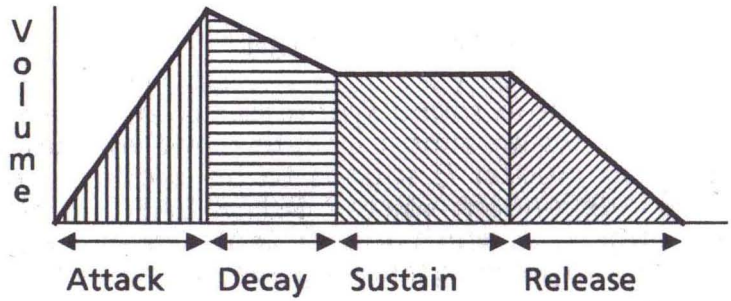
```
VOLUME 15
```

This will select maximum volume for all three voices. Because of the design of the SID chip it is not possible to select the volume of each voice independently using the **VOLUME** command.

SHAPE

After setting the volume, the *envelope* of the sound must be set for each voice using the **SHAPE** command. The envelope controls the way in which

the volume of sound on a given voice rises and falls with time, as shown in the diagram below:



The Envelope of a sound

The **SHAPE** command has five parameters, specifying the *Voice* (1, 2 or 3) and the *Attack*, *Decay*, *Sustain* and *Release* numbers (0 to 15) for the envelope required. The value used for Sustain controls the volume of the note after the Attack and Decay phases. A sustain of 15 will cause the note to sound at 100% of the volume set with the volume command, a value of 5 will sustain at about 30% of the maximum volume etc. The durations of Attack, Decay and Release for the permitted range of values are shown opposite.

Type in the following example of the **SHAPE** command:

```
SHAPE 1,8,10,15,15
```

This will set up the envelope for voice 1 with the following characteristics:

Attack = 8 = 100ms

Decay = 10 = 1.5s

VALUE	ATTACK	DECAY	RELEASE
0	2 ms	6 ms	6 ms
1	8 ms	24 ms	24 ms
2	16 ms	48 ms	48 ms
3	24 ms	72 ms	72 ms
4	38 ms	114 ms	114 ms
5	56 ms	168 ms	168 ms
6	68 ms	204 ms	204 ms
7	80 ms	240 ms	240 ms
8	100 ms	300 ms	300 ms
9	250 ms	750 ms	750 ms
10	500 ms	1.5 s	1.5 s
11	800 ms	2.4 s	2.4 s
12	1 s	3 s	3 s
13	3 s	9 s	9 s
14	5 s	15 s	15 s
15	8 s	24 s	24 s

Attack, Decay and Release parameters

Sustain = 15 = 100% of max. volume

Release = 15 = 24s

GATE

When a noise is being output on any voice the Attack, Decay and Sustain cycles will occur automatically since they are controlled by the SID chip but the Release cycle has to be initiated by your program. The **GATE** command is used to start the Release cycle. It has only one parameter – the number of the voice to be affected. For example:

```
GATE 1
```

You will need to use the **GATE** command to hear the full effects of the **SHAPE** parameters on noises you make later.

Having set up an envelope for voice one, the next step is to create a noise - using the **SOUND** command.

SOUND

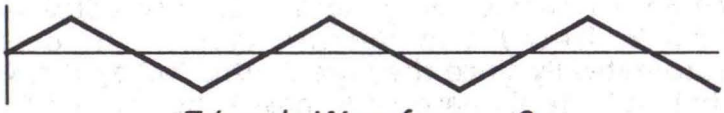
The **SOUND** command allows you play a tone on any voice, it has three parameters specifying the voice (1, 2 or 3), the frequency of the sound to be produced (0 to 65535) and the type of waveform to be used (0 to 3).

The frequency of the note produced is governed by the frequency number in the **SOUND** command according to the equation:

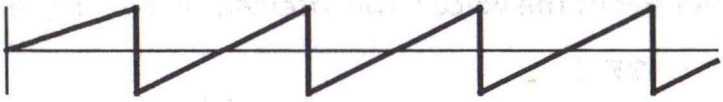
Frequency = Frequency number * 0.06 Hz.

(This will be of little interest unless you particularly want to generate notes of a certain frequency).

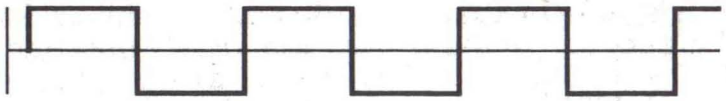
The Waveform parameter allows you to choose from the four different waveforms the SID chip can generate. These are illustrated below:



Triangle Waveform = 0



Sawtooth Waveform = 1



Pulse Waveform =2

Noise Waveform =3

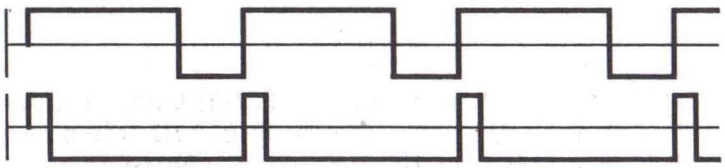
Try typing the following example:

```
SOUND 1,5000,1
```

If you set up the volume and envelope as described earlier this will play a note on voice 1 having a frequency of (5000*0.06 Hz.) using a sawtooth waveform. The sawtooth waveform is quite a harsh sound; try changing the waveform parameter to 0, the Triangle waveform, for a much smoother flute-like noise.

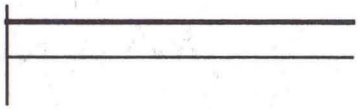
The fourth waveform, Pulse, is slightly more complicated than the others in that it requires an extra parameter to the **SOUND** command.

It is possible to alter the proportions of the wave by altering the width of the peaks, to produce a variety of alternative waveforms.

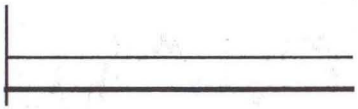


Alternative Pulse Waveforms

This is done by specifying the width of the peak as a proportion of the whole cycle. The proportion is set by choosing a pulse width parameter between 0 and 4095. A value of 0 or 4095 would give a straight line:

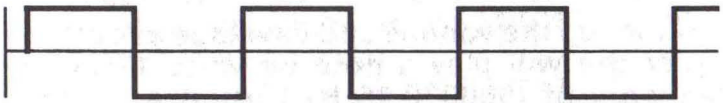


Pulse width 4095



Pulse width 0

A value of 2047 would make the peak width 50% of the total, giving:



Pulse Width = 2047 (50%)

The characteristics of the noises produced by the **SOUND** command will depend not only on the waveform you choose but on the envelope you set up with the **SHAPE** command. Try experimenting with the two commands together.

During your experiments with the **SOUND** command you will probably want to turn off the noise at times – this is achieved with the **QUIET** command.

QUIET

The **QUIET** command has no parameters and will turn off any output from the SID chip and completely reset all the registers. Note that you

will need to set up the volume and envelope after a **QUIET** command has been used.

MULTIPLE VOICES

When you are experimenting with the **SOUND** command, remember that there are three independent voices to use – each of which can have totally different envelopes and waveforms as well as frequencies.

One of the more interesting things you can do with the SID chip is create music, and SUPER BASIC provides a command to make this process as easy as possible; the **MUSIC** command.

MUSIC

The **MUSIC** command performs the frequency calculations necessary to play tunes across a span of eight octaves. It has four parameters:

Voice	(1, 2 or 3)
Octave	(0 to 7)
Note	(0 to 11)
Waveform	(0 to 3 + pulse width as required)

The **MUSIC** command is similar to the **SOUND** command except that frequency is calculated from the Octave and Note parameters. The range of octaves and notes available is detailed in Appendix M of the Commodore 64 Users Guide. The table overleaf shows the relation between notes and the **MUSIC** command note parameter.

PARAMETER	NOTE
0	C
1	C#
2	D
3	D#
4	E
5	F
6	F#
7	G
8	G#
9	A
10	A#
11	B

The Note parameters

The following short program demonstrates the **MUSIC** command and the range of the SID chip by playing a few scales:

```
10    REM SCALES
20    VOLUME 10
30    SHAPE 1,10,8,15,9
110   FOR O=0 TO 7
120   FOR N=0 TO 11
130   MUSIC 1,O,N,1
140   PAUSE 60
150   NEXT N:NEXT O
160   QUIET
```

The next program plays a few bars from the National Anthem in 3 part harmony. Notice how the sustain parameters for voices 2 and 3 are lower than that for voice 1 to allow the melody line (played on voice 1) to dominate the sound.

```
10    REM NATIONAL ANTHEM
20    VOLUME 10
```

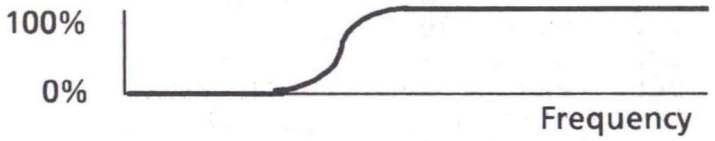
```
30    SHAPE 1,0,0,15,0
40    SHAPE 2,0,0,10,0
50    SHAPE 3,0,0,10,0
60    REM
100   READ O1,N1,O2,N2,O3,N3,T
110   IF O1=99 THEN 200
120   MUSIC 1,O1,N1,1
130   MUSIC 2,O2,N2,1
140   MUSIC 3,O3,N3,1
150   PAUSE T*15
160   GATE 1:GATE 2:GATE 3
170   GOTO 100
200   QUIET:END
900   REM OCTAVE & NOTE DATA
1000  DATA 4,7,3,11,1,7,2
1010  DATA 4,7,3,11,2,4,2
1020  DATA 4,9,4,4,2,0,2
1030  REM ********************
1040  DATA 4,6,4,2,2,2,3
1050  DATA 4,7,4,2,2,4,1
1060  DATA 4,9,4,2,2,6,2
1070  REM ********************
1080  DATA 4,11,4,2,2,7,2
1090  DATA 4,11,4,4,2,4,2
1100  DATA 5,0,4,9,2,0,2
1110  REM ********************
1120  DATA 4,11,4,2,2,2,3
1130  DATA 4,9,4,0,2,3,1
1140  DATA 4,7,3,11,2,4,2
1150  REM ********************
1160  DATA 4,9,4,0,2,0,2
1170  DATA 4,7,3,11,2,2,2
1180  DATA 4,6,3,9,1,2,2
1190  DATA 4,7,3,11,1,7,2
1200  DATA 99,0,0,0,0,0,0
```

You will need to experiment to create waveforms which sound like particular musical instruments but some very pleasing effects are possible with a little effort.

One command which may prove helpful in modifying the SID chip output to produce that elusive wailing sax is **FILTER**.

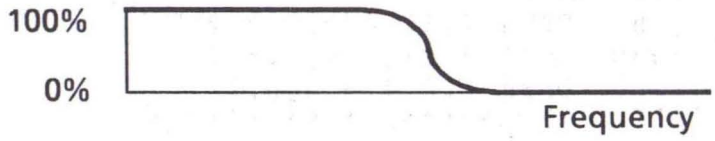
FILTER

The SID chip has a built-in filtering facility which filters the output from all three voices together. Three different filters may be used, either singly or in combination. The three are a High-pass filter,

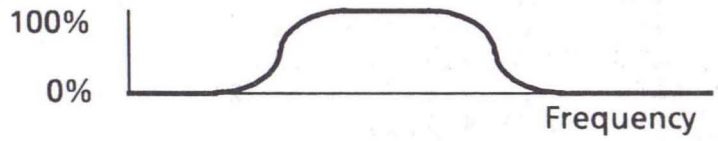


High-Pass Filter

which allows high frequency sounds to be output but suppresses the lower frequencies, a Low-pass filter which suppresses high frequencies, and a

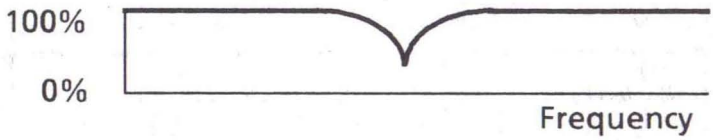


Low-Pass Filter

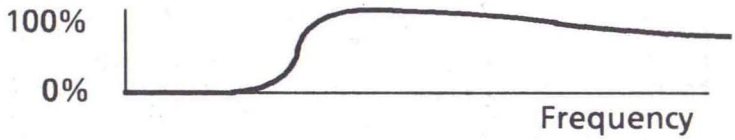


Band-Pass Filter

Band-pass filter which allows only the frequencies within a limited range to pass. Two filters may be combined to produce more complex filtering.



High-Pass and Low-Pass Filters Combined



High-Pass and Band-Pass Filters Combined

The **FILTER** command has four parameters:

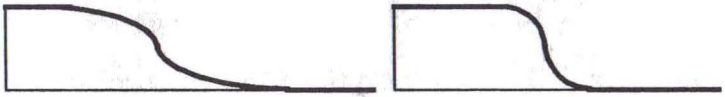
Voice	0-7 selecting which voice or voices are to be filtered (see table overleaf).
Filter type	0-7 selecting the filter or combination of filters to be used.
Frequency	0-2048 selecting the cut-off frequency of the filter.
Resonance	0-15 selecting the resonance or rate of cut-off of the flter.

The values of the Voice and Filter parameters are given in the table overleaf.

The frequency parameter is a number proportional to the actual cut-off frequency – 0 being the lowest cut-off frequency the SID chip can handle (about 30 Hz) and 2047 being the highest (about 40 kHz). In practice the actual values don't matter too much – experimenting is the best way to find the sound you want.

No.	Voice(s)	Filter Type
0	OFF	OFF
1	1	LOW
2	2	BAND
3	1 & 2	LOW & BAND
4	3	HIGH
5	1 & 3	HIGH & LOW
6	2 & 3	HIGH & BAND
7	1 & 2 & 3	LOW & BAND & HIGH

The resonance value controls the sharpness of the cut-off. The higher the number, the sharper is the filter cut-off of the unwanted frequencies.



Low and High Resonance Filters

SUMMARY - SOUND

SOUND plays a tone of a specified frequency and waveform using the selected voice.

```
SOUND Voice,Freq,Wave,(PW)
```

MUSIC plays one of twelve notes from one of eight octaves using the specified waveform and voice.

```
MUSIC Voice,Oct,Note,Wave,(PW)
```

VOLUME sets the volume for all three voices.

```
VOLUME Vol
```

SHAPE sets the Attack, Decay, Sustain and Release parameters for the specified voice.

```
SHAPE Voice,Att,Dec,Sus,Rel
```

GATE turns off the Gate bit of the specified voice, causing the release cycle of the waveform to commence.

```
GATE Voice
```

FILTER selects the type of filter or filters to be used, the cutoff frequency and resonance values, and which voice or voices are to be filtered.

```
FILTER Voice,Type,Freq,Reson
```

QUIET resets the SID chip.

```
QUIET
```

GRAPHICS COMMANDS

SUPER BASIC includes eleven commands for controlling the graphics of the Commodore 64. The 64 has two types of display; the normal text screen which shows 25 lines of 40 characters and uses 1000 bytes of memory, and the two bit-mapped displays which use 8000 bytes of memory and have individually addressable pixels.

TEXT MODE COMMANDS

There are three commands which alter the colours of the text display.

PAPER

The **PAPER** command sets the background colour of the text display. The syntax is:

```
PAPER colour
```

The colour parameter may be any number from 0 to 15. The colours associated with the numbers are shown in the table on the next page.

INK

The second command, **INK**, controls the colour of the letters printed on the text mode display. The choice of colours is the same as for **PAPER**.

Using **INK** has the same effect as using **PRINT**ing control characters, but has the advantage of making listings easier to read, and allows variables to be used to hold the colour of the text to be displayed.

NUMBER	COLOUR
0	Black
1	White
2	Red
3	Cyan
4	Purple
5	Green
6	Blue
7	Yellow
8	Orange
9	Brown
10	Light Red
11	Grey 1
12	Grey 2
13	Light Green
14	Light Blue
15	Grey 3

FRAME

FRAME sets the colour of the border area around the display using the same codes as **INK** and **PAPER**. This command works in both text and high resolution display modes.

HIGH RESOLUTION COMMANDS

There are eight commands in SUPER BASIC which control high resolution graphics.

CHIRES

This command clears the high resolution screen, resets the colour memory areas, and switches the display to two colour high resolution mode

(standard bit-mapped mode). There are no parameters needed, so you just type:

```
CHIRES
```

CMULTI

This command is similar to **CHIRES**, but switches to multicolour bit-mapped mode.

The text screen is not cleared by the **CHIRES** or **CMULTI** commands.

TEXT

This command returns the display to text mode, without clearing either the text display or the high resolution display. The display will also return automatically to text mode if there is a syntax error in a command or program, so that you can read the error message.

HIRES

This switches to the high-resolution mode last used, but does not clear the screen. This means you can swap between text and graphics displays without having to redraw or reprint the display each time.

SETCOL

The high resolution display modes use *colour registers* to control the colours used by the graphics commands. The normal high resolution mode uses two colours. The background colour is referred to as colour 0, and the foreground colour is colour 1. The colours to be used are set by **SETCOL**, which has two parameters:

Colour number (foreground/background)

Colour value (see the table above)

So, the command

```
SETCOL 0,1
```

sets the background colour to white, while

```
SETCOL 1,2
```

sets the foreground colour to red.

Multicolour high-res mode is more complicated. This has four colours, which are numbered from 0 to 3.

```
SETCOL 3, colour
```

sets the background colour; colours 0, 1 and 2 are foreground colours. You will notice that changing the background colour of the multicolour display alters the background of the text screen as well, and setting colour 2 changes the ink colour. This is because of the way the VIC chip stores colour data, and cannot be avoided.

PLOT

This command plots and erases points on the screen in both normal and multicolour high resolution modes. The command has three parameters:

X coordinate of the point to be plotted.

Y coordinate.

Colour register number.

so the syntax of the command is:

```
PLOT X,Y,col
```

The colour number indicates which of the preset colours is to be used. If you set this parameter to the number of the background, the point will be erased.

In standard bit-mapped mode, the colour number is the same as that used by **SETCOL** – background is 0, foreground is 1. In multicolour mode things are more complicated. The table below gives details of which **SETCOL** number corresponds to which **PLOT** number.

SETCOL number	PLOT number	
0	2	Foreground
1	1	Foreground
2	3	Foreground
3	0	Background

The command works in both normal and multicolour high-res mode. The normal high-res screen is 320 pixels by 200, so the coordinates allowed are:

X 0 to 319

Y 0 to 199

In multicolour mode the range of Y is the same, but the number of pixels horizontally is reduced to 160, so the range across is:

X 0 to 159

If your programs use coordinates larger than these (or less than zero), SUPER BASIC will switch back to text mode and print the message:

```
ILLEGAL QUANTITY ERROR IN XXXX
```

DRAW

DRAW allows you to draw lines on the high resolution displays. The line is drawn from the last point to be plotted to the point specified after the command. For example:

```
DRAW 100, 50,1
```

would draw a line ending at X = 100, Y = 50. The third parameter is the colour register number to be used (this is the same as in the **PLOT** command). For full control of the **DRAW** command you need to specify the start of the line with a **PLOT** command.

```
PLOT 10, 10, 1

DRAW 20, 20, 1
```

would draw a line from 10, 10 to 20, 20.

If you want to join a second line to the end of the first one, you don't need to **PLOT** the start of the second line. This means you can draw shapes with one **PLOT** command to start, and then one **DRAW** command for each of the sides. The command sequence:

```
PLOT 10, 10, 1

DRAW 70, 10, 1

DRAW 70, 50, 1
```

```
DRAW 10, 50, 1

DRAW 10, 10, 1
```

draws a rectangle sixty pixels by forty.

The coordinate ranges allowed in the two graphics modes are described in the section on the **PLOT** command. If you try to use too large a value for X or Y, the 64 will switch back to text mode and display an error message.

BLOCK

This command fills rectangles of colour on the graphics mode displays. There are five parameters:

The coordinates of the top left corner of the rectangle to be filled

The coordinates of the bottom right corner

The colour number

So the command:

```
BLOCK 10, 10, 70, 50, 1
```

would fill the rectangle outlined by the sequence given above.

The commands **PLOT**, **DRAW** and **BLOCK** will draw on the graphics screen memory even if the 64 is displaying text at the time. This means that you can create a graphics picture without showing it being built up, and the whole picture will appear instantly when you switch back to graphics with the **HIRES** command. (You must of course have set one or other of the high resolution modes with **CHIRES** or **CMULTI** before doing this.) This feature

also allows you to try ideas at the keyboard and still see what you are typing.

GRAPHICS EXAMPLE

This short program will show you the possibilities of the SUPER BASIC graphics commands.

```
90      REM  GRAPHICS DEMO
100     CMULTI
110     SETCOL 3,1: SETCOL 1,10
120     SETCOL 2,14: SETCOL 0,3
130     REM DRAW RANDOM CUBES
140     X = RND(0)*131
150     Y = RND(0)*171+8
160     C = RND(0)*3+1
170     GOSUB 1000
180     GET K$: IF K$="" THEN 130
190     REM CHANGE COLOURS
200     SETCOL RND(0)*4, RND(0)*16
210     GET K$: IF K$="" THEN 200
220     TEXT: END
990     REM DRAW A CUBE
1000    PLOT X, Y, C
1010    DRAW X+8, Y-8, C
1020    DRAW X+28, Y-8, C
1030    DRAW X+28, Y+12, C
1040    DRAW X+20, Y+20, C
1050    PLOT X+20, Y, C
1060    DRAW X+28, Y-8, C
1070    BLOCK X, Y, X+20, Y+20, C
1080    RETURN
```

HOW THE PROGRAM WORKS

100	Set multicolour mode.
110–120	Set up all four colour registers.
140–150	Pick random X and Y coordinates for start of cube drawing.

160	Pick random colour to draw cube.
170	Draw cube and fill in one face in chosen colour.
180	If no key pressed draw another cube somewhere else, otherwise:
200–210	Cycle all four colour registers at random with random colours until a key is pressed.
220	Return to text mode.
1000	Plot point at X,Y in colour C.
1010–1060	Draw a cube.
1070	Fill one face of the cube.

HOW THE DISPLAYS WORK

The text screen normally displayed by the Commodore 64 uses 1000 bytes of memory to hold the 25 rows of 40 characters. The memory area for this display begins at location 1024 and extends as far as 2023. A second block of memory from 55296 to 56295 holds information about the colour of each character on the display. This colour memory is outside the BASIC RAM area and does not affect BASIC program storage.

The high resolution display modes need much more memory than the text display; 8000 bytes for the screen map, and another 1000 bytes for storing colour information. This area of memory must be taken from the BASIC program storage area. Unfortunately, because of the way in which the VIC chip addresses the memory, all 9000 bytes must be in the same 16k quarter of the total memory. SUPER BASIC graphics displays use the memory from location 23568 to 32767, and so the amount of memory available for BASIC programs is reduced to about 21k. You should reset the top of

BASIC memory to 23568 (use the **HIMEM** command) before using the graphics displays from within a program, or the displays may be corrupted by the BASIC variables, which are stored in the top part of the BASIC memory area.

This memory arrangement leaves an 8k space at the top of the BASIC program area from 32768 to 40959 (there is not quite enough space in this quarter of the memory for the high resolution displays). This cannot be used automatically by the BASIC programs, but you could store machine code programs here.

SUMMARY - GRAPHICS

The SUPER BASIC graphics commands are:

PAPER Sets the background colour of the text screen.

```
PAPER colour
```

INK Changes the character colour of the text screen.

```
INK colour
```

FRAME Changes the border colour in both display modes.

```
FRAME colour
```

CHIRES Clears the high-resolution screen and switches to bit-mapped display mode.

```
CHIRES
```

CMULTI Clears the high-resolution screen and switches to multi-colour bit-mapped mode.

```
CMULTI
```

TEXT Returns to text mode.

```
TEXT
```

HIRES Returns to the last selected high-resolution mode without clearing the screen.

```
HIRES
```

SETCOL Controls the colours of bit-mapped displays.

```
SETCOL register, colour
```

PLOT Sets and clears individual pixels on the high-resolution displays.

```
PLOT X, Y, colour
```

DRAW Draws lines on the high-resolution display.

```
DRAW X, Y, colour
```

BLOCK Fills rectangular areas with colour in high-resolution modes.

```
BLOCK X1, Y1, X2, Y2, colour
```

SPRITE COMMANDS

SUPER BASIC contains six commands which make programs using sprites easier to write and understand.

SPRITE and SPROFF

These two commands turn sprites on and off. They have only one parameter – a value 0 to 7 – which selects which of the eight possible sprites is to be controlled. For example:

```
SPRITE 0
```

would turn on sprite 0, and:

```
SPROFF 3
```

would turn off sprite 3.

SPRMOV

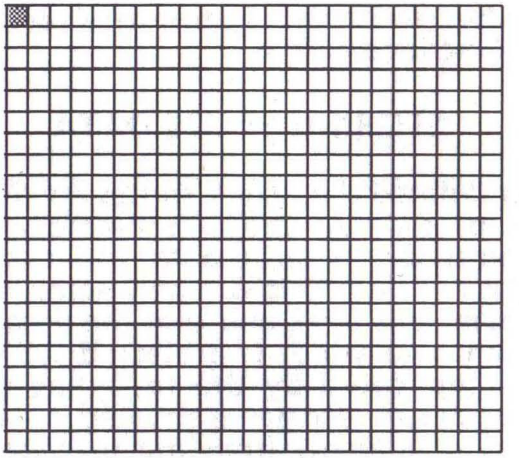
The position of each sprite on the screen is controlled by the **SPRMOV** command. **SPRMOV** has three parameters – the number of the sprite involved and its X and Y coordinates.

```
SPRMOV N, X, Y
```

The X and Y coordinates refer to the top left hand corner of the 24 by 21 pixel grid defining a sprite as shown on the next page.

For example:

```
SPRMOV 1,152,150
```



The Sprite Grid showing coordinate reference point (shaded)

would place the top left hand corner of sprite 1 in the centre of the screen. The coordinates used are not the same as the screen coordinates, as the sprite coordinates extend beyond the edges of the display to allow you to move the sprites smoothly on and off the screen. The valid parameters for **SPRMOV** are:

X	0 - 511
Y	0 - 255

The following short program will move sprite 1 from one side of the screen to the other:

```
5     SPRITE 1
10    FOR X=0 TO 400
20    SPRMOV 1,X,100
30    NEXT X
```

If you tried the last example you'll have noticed that the sprite was an unrecognisable blob. This is because the 64 needs to know the address of the block of memory which defines the sprite. The

way in which sprite data is stored for use by SUPER BASIC is described below.

STORING SPRITE DATA

Like user defined characters, sprites are defined by a block of data in the 64's memory. When allocating space for sprite data, you must bear in mind that the VIC II chip can only access one bank of 16k bytes at a time, and that the data must be kept in an area which is protected from BASIC. This is complicated by the fact that the high resolution graphics screen is situated in a different 16k block of memory from the text screen (see GRAPHICS COMMANDS Chapter), so that a block of data for a sprite on a text display must be moved into the same 16k area as the high resolution screen if that sprite is to be displayed on a high resolution display.

The **SPRSET** command is used to define this data area and the type of sprite required.

SPRSET

The **SPRSET** command has five parameters defining the type (standard or multicolour), colour, priority and data area of each sprite. The values are:

Type	0 for standard 1 for multicolour
Colour	0–15
Priority	0 Sprite in front of graphics 1 Sprite behind graphics
Data Area	The address of the sprite data block.

So the command:

```
SPRSET 5, 0, 2, 1, 832
```

defines sprite 5 to be standard (one-colour), coloured red, with low priority, and with the sprite data beginning at location 832.

SPRCOL

Multicolour sprites can have three colours. The first colour is specified separately for each sprite by the **SPRSET** command, but the second and third colours must be the same for all multicolour sprites. The two colours to be used are controlled by two colour registers which are set by **SPRCOL**. The command has two parameters:

Colour register number (0 or 1)

Colour (0–15)

So the command:

```
SPRCOL 0,6
```

would set the first of the multicolour sprite colour registers to blue.

SPRSIZ

Sprite sizes may be increased from normal to double width, double height or both by using this command. There are three parameters:

The sprite number

The width – 0 for normal, 1 for double size

The height – 0 for normal, 1 for double size

So to double the width and height of Sprite 0 type:

```
SPRSIZ 0, 1, 1
```

SPRITE EXAMPLE

This program displays all eight sprites as multicoloured faces.

```
1     REM MULTICOLOUR SPRITES
50    PRINT "{CLS}"
60    FOR S=0 TO 7
70    SPRMOV S, 0, 0
80    NEXT S
100   FOR Z=0 TO 63
110   READ D
120   POKE 832+Z, D
130   NEXT Z
140   SPRCOL 0,2: SPRCOL 1,1
150   FOR S=0 TO 7
160   SPRITE S
170   SPRSET S, 1, 4, 1, 832
180   SPRSIZ S, 0, 0
190   X=25
200   FOR Y=50 TO 200 STEP 4
210   SPRMOV S, X, Y
220   X=X+S
230   NEXT Y
240   NEXT S
300   PAUSE 300
310   FOR S=0 TO 7
320   SPROFF S
330   NEXT S
1000  DATA 8, 136, 136, 42, 170
1010  DATA 168, 42, 170, 168, 37
1020  DATA 170, 88, 148, 105, 22
1030  DATA 165, 170, 90, 170, 170
1040  DATA 170, 170, 170, 170, 42
1050  DATA 40, 168, 42, 40, 168
1060  DATA 170, 170, 170, 175, 255
1070  DATA 250, 181, 85, 94, 181
```

```
1080  DATA 85, 94, 173, 85, 122, 43
1090  DATA 85, 232, 42, 255, 168
1100  DATA 10, 170, 160, 2, 170
1110  DATA 128, 0, 170, 0
1120  DATA 0, 40, 0, 0
```

HOW THE PROGRAM WORKS

60–80 Position all 8 sprites out of site at the top left corner of the screen.

100–130 READ sprite data and POKE into the cassette buffer starting at location 823. Since display is text mode the cassette buffer is in the same 16k block of memory.

140 Set sprite colour registers, then for each sprite:

160 Switch on sprite.

170 Set Multicolour, Purple, Low priority, data area starts at 832.

180 Set standard size.

190–240 Move diagonally to bottom of screen.

250–330 Wait 5 seconds then turn all sprites off.

SUMMARY - SPRITES

SPRITE Switches on a sprite.

```
SPRITE number
```

SPROFF Switches sprites off.

```
SPROFF number
```

SPRMOV Sets positions of sprites on the screen.

```
SPRMOV number, X, Y
```

SPRSET Sets sprite properties.

```
SPRSET number, type, colour,
       priority, data
```

SPRCOL Controls colours of multicoloured sprites.

```
SPRCOL register, colour
```

SPRSIZ Adjusts sprite sizes.

```
SPRSIZ number, X size, Y size
```

SUPER BASIC COMMANDS

UTILITIES	AUTO	Start, Increment (Optional)
	DEL	Start, End
	RENUM	Start, Increment (Optional)
	PACK	
	DOKE	Address, Data
	DEEK	(Address)
	AT	X coordinate, Y coordinate
	PAUSE	Delay in seconds
	POP	
	OLD	
	HIMEM	Address
	RESET	
GRAPHICS	INK	Colour (0-15)
	PAPER	Colour (0-15)
	FRAME	Colour (0-15)
	CHIRES	
	CMULTI	
	TEXT	
	HIRES	
	SETCOL	Register, Colour
	PLOT	X coord, Y coord, Register
	DRAW	X coord, Y Coord, Register
	BLOCK	X1, Y1, X2, Y2, Register
SPRITES	SPRITE	Sprite (0 - 7)
	SPROFF	Sprite (0 - 7)
	SPRCOL	Register, Colour
	SPRSET	Sprite,Type, Colour, Priority, Data
	SPRMOV	Sprite, X coordinate, Y coordinate
	SPRSIZ	Sprite, Xsize, Ysize
SOUND	VOLUME	Volume (0-15)
	SHAPE	Voice, Attk, Decay, Sust, Rel
	SOUND	Voice, Frequency, Wave, (PW)
	GATE	Voice
	MUSIC	Voice, Octave, Note, Wave, (PW)
	QUIET	
	FILTER	Voices, Type, Freq, Resonance

SUPER BASIC COMMANDS

UTILITIES	AUTO	Start, Increment (Optional)
	DEL	Start, End
	RENUM	Start, Increment (Optional)
	PACK	
	DOKE	Address, Data
	DEEK	(Address)
	AT	X coordinate, Y coordinate
	PAUSE	Delay in seconds
	POP	
	OLD	
	HIMEM	Address
	RESET	
GRAPHICS	INK	Colour (0-15)
	PAPER	Colour (0-15)
	FRAME	Colour (0-15)
	CHIRES	
	CMULTI	
	TEXT	
	HIRES	
	SETCOL	Register, Colour
	PLOT	X coord, Y coord, Register
	DRAW	X coord, Y Coord, Register
	BLOCK	X1, Y1, X2, Y2, Register
SPRITES	SPRITE	Sprite (0 - 7)
	SPROFF	Sprite (0 - 7)
	SPRCOL	Register, Colour
	SPRSET	Sprite,Type, Colour, Priority, Data
	SPRMOV	Sprite, X coordinate, Y coordinate
	SPRSIZ	Sprite, Xsize, Ysize
SOUND	VOLUME	Volume (0-15)
	SHAPE	Voice, Attk, Decay, Sust, Rel
	SOUND	Voice, Frequency, Wave, (PW)
	GATE	Voice
	MUSIC	Voice, Octave, Note, Wave, (PW)
	QUIET	
	FILTER	Voices, Type, Freq, Resonance

THE COPY OF THIS SUMMARY ON THE NEXT PAGE MAY BE REMOVED TO FORM A CONVENIENT REFERENCE CARD

This pack contains:

- **a cassette with SUPER BASIC and some demonstration programs**
- **a booklet containing full documentation and examples of all the commands**

SUPER BASIC is a package of machine code routines which adds 36 new BASIC commands to your Commodore 64. The range of easy to use commands has been carefully chosen to enable the user to make maximum use of the extensive sound, graphics and sprite capabilities of the 64 and to ease the task of programming in BASIC by providing several useful utilities.

The booklet describes the commands in detail and illustrates their use with example programs. Example programs are also included on the cassette.

Key Features:

*** UTILITIES**

The 12 utilities provide AUTO line numbering, DELeting lines, RENUMbering, OLD (restores programs), HIMEM, DEEK, DOKE, PACK, AT, PAUSE, POP and RESET.

*** GRAPHICS**

The 11 graphics commands set the INK, PAPER and FRAME colours, HIRES, colour CHIRES, MULTIcoloured highres or TEXT screens, you may SETCOLS, PLOT, DRAW and BLOCK.

*** SPRITES**

The 6 sprite commands can be used to set up sprites (SPRSET), switch them on/off (SPRITE/SPROFF), colour them (SPRCOL), move them (SPRMOV) and alter their size (SPRSIZ).

*** SOUND**

The 7 superb sound commands enable easy setting of VOLUME, SHAPE, SOUND, MUSICAL notes, timing (GATE), frequency FILTERS and QUIET.

ISBN 0-7126-0666-1

9 780712 606660

£9.95

ISBN 0 7126 0666 1
CENTURY SOFTWARE

SUPER BASIC FOR THE COMMODORE 64

PETER LUPTON AND FRAZER ROBINSON